그룹통신 메시징 시스템의 메시지 순서를 이용한 멀티 데이터베이스에서 전역 동시성 제어 기법

정회원 문 애 경*, 남 궁 한* 조 행 래**

Global Concurrency Control using Message Ordering of Group Communication Messaging System in Multidatabase

Aekyung Moon*, Han Namgoong*, Haengrae Cho** *Regular Members*

요 약

멀티 데이터베이스 시스템(MultiDataBase System: MDBS)이란 기존에 독자적으로 개발되어 사용되고 있는 서로 다른 지역 데이터베이스 시스템(Local DataBase System: LDBS)들을 통합하기 위해 제안된 시스템으로 논리적으로 단일화된 인터페이스를 제공함으로써 산재된 정보의 효율적 통합 기능을 제공한다. 본 논문에서는 LDBS의 논리적인 통합 인터페이스로 메시징 시스템을 이용하고 메시지 순서 기능을 적용한 전역 동시성 제어 기법을 제안한다. 전역 동시성 제어 기법은 전역 직렬성을 보장하기 위하여 전역 트랜잭션의 서브 트랜잭션이 실행되는 모든 LDBS에서 같은 순서로 직렬화되어야 한다. 메시지 순서 기능은 하나의 노드가 여러 개 메시지를 송신하는 경우, 모든 노드는 송신한 순서에 따라 메시지 수신을 보장하는 것으로 이를 이용하여 전역 트랜잭션의 서브 트랜잭션들을 실행 노드에 같은 순서로 전송하고 실행 LDBS는 해당 순서대로 직렬화한다면 전역 직렬성을 보장할 수 있다. 그 결과 제안된 전역 동시성 제어 기법은 LDBS의 실행 정보 없이 전역 트랜잭션의 상대적인 실행 순서를 결정할 수 있기 때문에 지역 자치성을 보장한다.

Key Words : Multidatabase; Global Concurrency Control; Message Ordering; Group Communication.

ABSTRACT

A multidatabase system (MDBS) is designed to provide universal access to distributed data across multiple autonomous, heterogeneous local database systems (LDBSs). In this paper, we propose a global concurrency control (GCC) algorithm in MDBS. The proposed GCC algorithm adopts the idea of message ordering in group communication messaging system. A necessary condition of GCC algorithm is that global transactions are serialized in the same order at all LDBSs that they execute. If all operations of a transaction are bundled in a single message and the message arrives at LDBSs in the same order using message ordering property, each LDBS can perform subtransactions in the same order. As a result, message ordering allows to determine easily the relative serialization order of global transactions, and then it can support a local autonomy without any information about the serialization order of transactions executing locally.

Ⅰ. 서 론

최근 웹 문서 검색이나 컨텐츠 서비스 등을 지원하기 위한 다양한 데이터 자원이 발생하면서, 정보 통합에 대한 많은 기술 개발이 진행되고 있다[1]. 정보 통합은 각각의 지역 시스템의 데이터 저장소에는 영향을 끼치지 않으면서, 다양한 정보 자원을 통합함으로써 새로운 정보 서비스 창출을 목적으로

* 한국전자통신연구원 (akmoon,nghan@etri.re.kr)
** 영남대학교 전자정보공학부 (hrcho@yu.ac.kr)
논문번호 : 030403-0915, 접수일자 : 2003년 9월 15일

한다. 이러한 시도에는 CALS의 IDE(Integrated Data Environment), PDM(Product Data Management), 메타 검색 엔진, 디지털 도서관 등의 초기 정보 통합 서비스와 EAI(Enterprise Application Integration), EIP(Enterprise Information Portal) 등의 비즈니스 정보 통합 서비스가 있다. 멀티 데이터베이스 시스템(MultiDataBase System: MDBS)은 여러 곳에 산재해있는 이질형 데이터를 통합하기 위해 제안된 통합 시스템이다. 통합하고자 하는 정보 자원들의 데이터를 실제로 저장하는 것이 아니라 각 정보 자원을 갖는 지역 데이터베이스 시스템(Local DataBase System: LDBS)들을 네트워크, 또는 인터페이스로 연결하고 이에 대해 질의를 분해, 전달 후 질의 결과를 통합하는 가상 통합 시스템이다.

MDBS는 하나 이상의 LDBS를 액세스하는 전역 응용을 처리할 수 있다. 그러나 일반적인 분산 데이터베이스 시스템 환경과 달리 통합되는 LDBS들이 서로 다른 질의 처리 능력을 제공할 뿐만 아니라, 지역 자치성 요구 사항 때문에 전역 트랜잭션 처리를 위한 상호 협조가 불가능하여 전역 직렬성을 만족하는 전역 동시성 제어 기법의 설계가 어렵다[15]. 기존의 전역 동시성 제어 기법은 "자치성을 보장하는 기법"과 "자치성을 부분적으로 위반한 기법"으로 나눌 수 있다. 자치성을 보장하는 기법은 전역 동시성 제어 기법을 구현할 때 MDBS로 통합되는 LDBS를 제한하거나 수정을 필요로 하지 않는다. 따라서 MDBS로 통합되는 과정에서 지역 자치성을 보장할 경우, LDBS의 기존 소프트웨어를 수정없이 사용할 수 있으므로 시스템 통합 과정이 용이하며 MDBS의 확장성이 증가한다. 하지만, Breitbart[3]와 Georgakopolos[11]가 동시성 저하와 높은 철회율을 자치성 보장 기법의 문제점으로 지적한 이후 대부분의 전역 동시성 제어 기법은 지역 자치성을 부분적으로 위반한 기법을 가정하고 있다. 즉, 지역 자치성으로 인하여 기존 분산 통합 시스템 환경과 달리 실행되는 LDBS의 정보를 알 수 없기 때문에 MDBS에서 전역 동시성 제어 기법의 구현은 복잡하고 낮은 성능을 갖는다.

최근 연구되고 있는 JMS(Java Message Service)와 같은 메시징 시스템은 응용 프로그램들 간 비동기식 통신 방법을 제공하는 것으로 기업간 응용 통합 인터페이스로 중요성이 증가하고 있다[7]. 본 논문에서는 LDBS와 MDBS 사이의 논리적인 통합 인터페이스로 메시징 시스템을 이용하고 그룹통신 메시징 시스템의 메시지 순서 기능을 적용한 전역 동시성 제어 기법을 제안한다. 전역 동시성 제어기법은 전역 직렬성을 보장하기 위하여 전역 트랜잭션의 서브 트랜잭션이 실행되는 모든 LDBS에서 같은 순서로 직렬화되어야 한다. 메시지 순서 기능은 하나의 노드가 여러 개 메시지를 송신하는 경우, 송신한 순서와 같이 모든 노드에서 메시지 수신을 보장하는 것으로 이를 이용하여 전역 트랜잭션의 서브 트랜잭션들을 실행 노드에 같은 순서로 전송하고 실행 LDBS는 해당 순서대로 직렬화 수행한다면 전역 직렬성을 보장할 수 있다. 그 결과 본 논문에서 제안하는 전역 동시성 제어 기법은 LDBS의 실행 정보 없이 전역 트랜잭션의 상대적인 실행 순서를 결정할 수 있기 때문에 지역 자치성을 보장한다.

본 논문의 구성은 다음과 같다. 2절에서는 관련 연구에 대해 살펴보고, 3절에서는 본 논문에서 제안하는 전역 동시성 제어 기법을 설명한다. 4절에서는 성능 평가를 위한 분산 시스템 모형을 제안하고 성능 분석을 위해 채택된 매개 변수를 설명한다. 5절에서는 제안된 성능 평가 모형을 바탕으로 수행한 성능 평가 결과를 비교하고, 끝으로 6절에서 결론을 맺는다.

Ⅱ. 관련 연구

본 절에서는 먼저 기존에 제안된 전역 동시성 제어 기법을 설명하고 메시징 시스템과 그룹통신 메시징 시스템의 메시지 순서 기능을 설명한다.

2.1 전역 동시성 제어 기법

지역 자치성을 보장하는 전역 동시성 제어 기법에는 낙관적 티켓 기법(Optimistic Ticket Method: OTM)[11]과 chain-conflicting serializability[23]가 있다. 두 기법은 전역 트랜잭션 사이에 발생하는 간접 충돌을 해결하기 위하여 강제적인 직접 충돌을 유발한다. 특히, OTM은 모든 전역 트랜잭션들이 티켓이라는 공통된 데이터를 수정하게 함으로써 직접 충돌을 유도하는 기법으로, 직접 충돌로 인하여 비직렬 스케줄을 발견하고 지역 동시성 제어기로 하여금 그 스케줄을 수행하지 않게 함으로써 전역 직렬성을 보장한다. 그러나 OTM의 경우 동시에 실행되는 트랜잭션 수가 많아지면 빈번한 티켓 충돌로 인해 낮은 동시성과 높은 철회율로 성능이 떨어질 수 있고, 전역 교착상태가 발생할 수 있다는 단점을 갖는다.

지역 자치성을 부분적으로 위반한 전역 동시성 제어 기법은 (a) MDBS에 포함될 수 있는 LDBS를 제한하는 기법[4,11], (b) LDBS의 실행 정보를 요구하는 기법[5], 그리고 (c) 전역 직렬성보다 완화된 정확성 기준을 사용하는 기법[2,9,16,17]으로 분류할 수 있다. 첫 번째에 포함되는 대표적인 기법은 암시적 티켓 기법(Implicit Ticket Method: ITM)을 들 수 있다. ITM은 LDBS가 2단계 로킹과 같은 엄격한(rigorous) 트랜잭션 실행만을 지원할 때 사용할 수 있는 기법이다. 두 번째에 속하는 기법으로 사이트 그래프 알고리즘이 있다. 사이트 그래프 알고리즘은 사이트 그래프에 순환이 없는 스케줄만 실행하는 방법으로, LDBS에서 실행 스케줄을 MDBS에 알리지 않으면 그래프에서 완료된 전역 트랜잭션의 에지를 삭제하기 어렵다. 세 번째에 포함되는 전역 동시성 제어 기법으로는 2단계 직렬성(2-Level Serializability: 2LSR)[17]이 있다. 2LSR의 경우 지역 스케줄은 모두 직렬화 가능하고 전역 단계에서 지역 트랜잭션을 제외한 전역 트랜잭션들만 직렬 스케줄인 경우 정확한 수행 결과로 판단한다. 이 경우 전역 직렬화 스케줄은 항상 2LSR을 만족하지만 2LSR이 모두 전역 직렬성을 만족하지는 않는다. 따라서 2LSR이 전역 직렬성을 만족하기 위해서는 액세스 데이터 패턴에 제한을 두어야 한다.

2.2 메시지 순서

메시징 시스템은 응용간 비동기식 통신 방법을 제공하는 것으로 기업간 응용 통합, 인터넷 기반 비즈니스 솔루션 그리고 J2EE의 메시징 서버 솔루션 등으로 각광 받고 있다. 최근 상용화된 메시징 시스템으로는 IBM의 MQSeries, Progess사의 SonicMQ, Fiorano의 FioranoMQ 그리고 썬 마이크로 시스템즈의 JMQ 등을 들 수 있다[10,19]. 메시지 순서 기능은 그룹통신 메시징 시스템이 제공하는 기능이지만 최근에는 일반적인 메시징 시스템에도 채택되는 기능이다[14,22].

메시지 순서 기능은 노드가 여러 개 메시지를 방송하는 경우, 모든 노드는 송신한 순서에 따라 메시지 수신을 보장하는 것이다. 예를 들면, 두 노드 N_1과 N_2가 다른 노드로부터 메시지 m과 m'를 전송받는다면, N_1과 N_2의 메시지 실행 순서는 모두 m → m' 혹은 m' → m임을 보장한다. 이러한 메시지 순서 기능은 트랜잭션의 순서를 미리 정하는 것과 같은 역할을 하므로 전역 트랜잭션의 실행 LDBS에 대한 상대적인 직렬 순서를 제어 할 수 있고 전역 교착상태 해결에 용이하다[12,13].

Ⅲ. 메시지 순서 기반 전역 동시성 제어 기법

본 절에서는 메시지 순서 기반 전역 동시성 제어 기법(Global Concurrency Control based on Message Ordering: GCC-M)을 제안한다. GCC-M은 지역 자치성을 보장하면서 LDBS의 실행순서를 보장하기 위하여 강제 직접 충돌을 사용한다. 또한 GCC-M은 전역 트랜잭션이 각 LDBS에 실행을 할당하기 전에 액세스하는 데이터 집합을 알고 있고 LDBS는 2단계 로킹을 지원한다고 가정한다.

3.1 자료 구조 및 가정

MDBS는 전역 트랜잭션 관리자(Global Transaction Manager: GTM), 전역 트랜잭션 에이전트(Global Transaction Agent: GTA) 그리고 메시지 순서 컴포넌트 (Message Ordering Component: MOC)로 구성된다. 그림 1은 LDBS들을 MOC 인터페이스를 이용하여 통합한 MDBS 구조를 나타낸다. GTM은 전역 트랜잭션을 수락하고 그들의 실행을 담당하는 역할을 한다. 전역 트랜잭션은 동시에 실행되는 여러 개의 서브 트랜잭션으로 구성되는데, 각 서브 트랜잭션은 하나의 LDBS에 저장된 데이터베이스를 액세스한다. GTM은 서브 트랜잭션 사이에 직접 충돌 여부를 검사하기 위하여 트랜잭션 테이블(Transaction Table: TTBL)과 메시지 큐(Message Queue: MQ)의 두 가지 정보를 갖는다.

전역 트랜잭션 T_k의 i번째 서브 트랜잭션이 T_k^i인 경우, TTBL은 $<RS_k^i, WS_k^i>$를 기록한다. RS_k^i는 T_k^i의 판독 데이터 집합이고 WS_k^i는 T_k^i의 갱신 데이터 집합이다. MQ[i]는 $LDBS_i$에서 완료되지 않은 서브 트랜잭션의 식별자를 기록한다. 만약 MQ[i]에 T_{k-1}^i가 T_k^i보다 먼저 등록되어 있다면 직렬화 순서가 $T_{k-1} \rightarrow T_k$임을 의미한다. MOC는 기존에 존재하는 LDBS에 통합된 인터페이스를 제공함으로 써 메시지 순서 기능을 담당한다. 특히 MOC의 송신자는 각 서브 트랜잭션에 단일화된 타임스탬프를 부여하고 MOC의 수신자는 자신의 지역 큐에 수신한 서브 트랜잭션을 저장하고 타임스탬프 순서에 따라 실행한다[20]. MOC_i는 $LDBS_i$에 존재하는 MOC의 수신자를 의미한다.

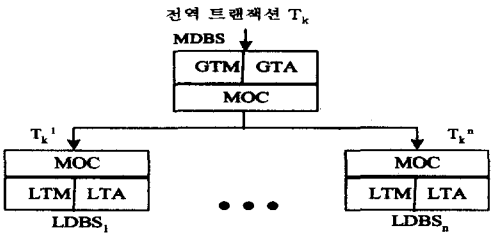


그림 5. MDBS 시스템의 구조

GTM은 T_k의 서브 트랜잭션들을 T_k에 할당된 GTA에 전달한다. 이후 GTA는 MOC를 통하여 각 서브 트랜잭션을 해당 LDBS에 전송한다. MOC는 모든 LDBS에서 동일한 순서로 실행되게 하기 위하여 메시지 순서에 따라 서브 트랜잭션을 실행한다. $LDBS_i$에는 MOC_i로부터 실행 순서대로 전송 받은 서브 트랜잭션을 병행 실행하기 위한 지역 트랜잭션 에이전트(Local Transaction Agent: LTA)가 존재한다. $LDBS_i$에서 LTA_k^i는 트랜잭션 T_k의 서브 트랜잭션을 실행하기 위한 LTA를 의미한다. LDBS는 전역 직렬성을 보장하기 위하여 실행 중인 서브 트랜잭션의 정보를 저장하는 실행 테이블(Execution Table: ETBL)을 유지한다. ETBL은 <트랜잭션 식별자, 충돌 연산, 실행 상태>의 정보를 갖는다. 서브 트랜잭션 T_k^i의 충돌 연산이 O_k^i인 경우, $O_k^i \in (RS_k^i \cup WS_k^i)$이고 O_k^i의 실행 상태는 $S(O_k^i)$로 표시한다. 단, $S(O_k^i)$의 초기 상태는 '0'이며, 실행 상태가 되면 '1'로 변경된다. 즉, 트랜잭션 T_k^i가 현재 실행 중이고 O_k^i가 현재 LTA_k^i에 의해 실행 요청되었다면 $S(O_k^i)$는 '1'이 된다. ETBL은 전역 트랜잭션의 실행을 제어하기 위하여 사용되는데 자세한 알고리즘은 3.2절에 설명한다.

3.2 GCC-M의 알고리즘

전역 트랜잭션들을 순차적으로 실행하더라도 지역 트랜잭션에 의해 발생하는 간접 충돌에 의해 전역 직렬성이 위반될 수 있다. 따라서 GCC-M은 간접 충돌을 해결하기 위하여 OTM과 같이 강제적인 직접 충돌 유발 방법을 사용한다. 그러나 OTM과 달리 전역 트랜잭션의 상대적인 직렬 순서를 사전에 알고 있기 때문에 불필요한 강제적인 직접 충돌을 제거할 수 있다. GCC-M의 구체적인 알고리즘은 다음과 같다.

단계 1: GTM은 전역 트랜잭션 T_k를 액세스하는 LDBS에 따라 서브 트랜잭션으로 분할한다. 단, 서브 트랜잭션 T_k^i는 $LDBS_i$를 액세스한다. 각 T_k^i에 대해 GTM은 $<RS_k^i, WS_k^i>$를 TTBL[k][i]에 등록하고, T_k^i를 MQ[i]에 등록한다.

단계 2: GTM은 MQ[i]에 저장된 바로 이전 전역 트랜잭션과의 관계를 검사한다. MQ[i]에 기록된 T_k^i의 이전 트랜잭션이 T_{k-1}^i라고 가정하자. 그러면 전역 트랜잭션의 직렬화 순서는 $T_{k-1} \rightarrow T_k$가 되어야한다. T_k^i의 갱신 데이터 집합 WS_k^i의 데이터 중 WS_{k-1}^i 혹은 RS_{k-1}^i에 속하는 데이터가 있으면 T_{k-1}^i와 T_k^i간에는 직접 충돌이 존재하고 충돌 정보는 $<O_{k-1}^i, O_k^i>$가 된다. 그러나 T_{k-1}^i와 T_k^i사이에 직접 충돌이 존재하지 않으면 다음과 같이 강제적인 충돌을 유발하기 위한 추가적인 연산이 필요하다.

(1) WS_{k-1}^i가 ∅이 아닌 경우, WS_{k-1}^i에 속하는 x를 선택하고 T_k^i의 마지막 위치에 R(x)를 추가한다. 이 때 충돌 정보는 $<W_{k-1}^i(x), R_k^i(x)>$이다.

(2) WS_{k-1}^i가 ∅인 경우, RS_{k-1}^i에 속하는 x를 선택하고 T_k^i의 마지막 위치에 W(x)를 추가한다. 이 때 x는 원본과 동일한 값을 갖는다. 충돌 정보는 $<R_{k-1}^i(x), W_k^i(x)>$이다.

단계 3: T_k의 모든 서브 트랜잭션에 대하여 충돌 정보(O_{k-1}^i, O_k^i)가 결정되면, GTM은 GTA에게 그들의 충돌 정보와 함께 T_k의 서브 트랜잭션을 전달한다. 그러면, GTA는 실행 LDBS의 MOC의 수신자에게 각각의 서브 트랜잭션을 전송한다.

단계 4: 서브 트랜잭션 T_k^i와 관련 충돌 정보를 전송 받으면, MOC_i는 T_k^i의 충돌 연산 (O_k^i)와 $S(O_k^i)$를 ETBL에 등록한다. $S(O_k^i)$의 초기 상태는 '0'이다. MOC_i는 LTA_k^i에게 서브 트랜잭션 T_k^i와 O_k^i를 전송한다. 그러면 LTA_k^i는 O_k^i를만날 때까지 T_k^i의 연산을 순서대로 실행 요청한다. 만약, 요청하고자 하는 연산이 O_k^i인 경우, LTA_k^i는 먼저 MOC_i를 통해 O_k^i와 충돌되는 연산이 실행되었는지 확인한다. MOC_i는 O_k^i의 충돌 연산 O_{k-1}^i의 $S(O_{k-1}^i)$를 LTA_{k-1}^i에게 확인한다. O_{k-1}^i가 속한 서브 트랜잭션이 아직 O_{k-1}^i의 실행을 요청하지 않았다면 즉, $S(O_{k-1}^i)$가 '0'이면 O_k^i는 해당 T_{k-1}^i의 O_{k-1}^i가 실행될 때까지 대기한다. 이 때 MOC_i와 $LDBS_i$사이에 지역 교착상태를 유발할 수 있다. O_k^i가 무한히 대기하는 것을 방지하기 위하여 타임아웃 기법을 사용한다. ■

다음 [예 1]은 전역 트랜잭션들이 순차적으로 실행되더라도 간접 충돌로 인하여 전역 직렬성이 위반된 경우와 전역 직렬성을 보장하기 위한 GCC-M

의 실행 과정을 보여준다.

[예 1] 지역 데이터베이스 시스템 $LDBS_1$에 데이터 a가 존재하고 $LDBS_2$에 데이터 b와 c가 존재한다고 가정하자. 전역 트랜잭션 G_1과 G_2의 내용은 아래와 같다.

$$G_1 : R_1(a)\ W_1(c)$$
$$G_2 : W_2(a)\ R_2(b)$$

$LDBS_2$에서 실행되는 지역 트랜잭션 T_3는 다음과 같다.

$$T_3 : R_3(c)\ W_3(b)$$

G_1과 G_2가 각 LDBS에서 순차적으로 실행되더라도 다음과 같은 잘못된 실행 스케줄이 발생할 수 있다.

$LSH(LDBS_1) : R_1(a)W_2(a)\quad (G_1 \rightarrow G_2)$

$LSH(LDBS_2) : R_3(c)W_1(c)R_2(b)W_3(b)$

$(G_2 \rightarrow T_3 \rightarrow G_1)$

즉, $LDBS_1$에서는 G_1이 G_2전에 직렬화되고 $LDBS_2$에서는 간접 충돌로 인하여 G_2가 G_1전에 직렬화된다. 그 결과 전역 직렬성은 위반되었다.

다음으로 GCC-M을 적용하여 위의 문제를 해결하는 과정을 설명한다. 단, 전역 트랜잭션은 G_1, G_2 순으로 발생하고 $LDBS_1$과 $LDBS_2$의 MQ는 ∅라고 가정한다. GTM은 전역 트랜잭션 G_1을 액세스하는 LDBS에 따라 서브 트랜잭션으로 분할하고 G_1^1의 $<RS_1^1=\{a\},\ WS_1^1=\varnothing>$을 TTBL[1][1], G_1^2의 $<RS_1^2=\varnothing,\ WS_1^2=\{c\}>$를 TTBL[1][2]에 각각 등록한다. 그리고 MQ[1], MQ[2]에 G_1을 등록하고 G_1 이전 전역 트랜잭션과의 관계를 검사한다. 현재, 모든 MQ에 G_1 이전 전역 트랜잭션이 없는 상태이므로 GTM은 G_1^1, G_1^2를 추가 연산 없이 GTA에 전달한다. 그러면 GTA는 송신 MOC를 통하여 G_1^1, G_1^2를 $LDBS_1$의 MOC의 수신자와 $LDBS_2$의 MOC의 수신자에게 각각 전달한다.

다음 GTM은 전역 트랜잭션 G_2를 액세스하는 LDBS에 따라 서브 트랜잭션으로 분할하고 G_2^1의 $<RS_2^1=\varnothing,\ WS_2^1=\{a\}>$를 TTBL[2][1]에 G_2^2의 $<RS_2^2=\{b\},\ WS_2^2=\varnothing>$을 TTBL[2][2]에 각각 등록한다. 이후 MQ[1], MQ[2]에 G_2를 등록하고 G_2 이전 전역 트랜잭션과의 관계를 검사한다. MQ[1]에 저장된 G_2의 이전 전역 트랜잭션은 G_1이다. 그 결과 $LDBS_1$의 전역 트랜잭션의 직렬화 순서는 $G_1 \rightarrow G_2$이다. 이 경우, $WS_2^1 \cap (RS_1^1 \cup WS_1^1) \neq \varnothing$이므로 G_1^1과 G_2^1사이에는 직접 충돌이 존재한다. 따라서 추가 연산이 필요하지 않다. 마찬가지로 MQ[2]에 저장된 G_2의 이전 전역 트랜잭션은 G_1이므로 $LDBS_2$에서의 전역 트랜잭션의 직렬화 순서도 $G_1 \rightarrow G_2$이다. 하지만, $WS_2^2 \cap (RS_1^2 \cup WS_1^2) = \varnothing$이므로 강제적인 충돌을 유발하기 위한 추가연산이 필요하다. 이때 $WS_1^2=\{c\}$이므로 단계 2-(1)에 따라 $R_2^2(c)$가 G_2^2에 추가된다. 그 결과 사전에 직접 충돌이 존재하는 $LDBS_1$에서는 추가적인 연산이 필요하지 않고 $LDBS_2$에서만 강제적인 직접 충돌이 필요하다. 지역 스케줄은 다음과 같다.

$LSH(LDBS_1)$: $R_1(a)W_2(a)$

$LSH(LDBS_2)$: $R_3(c)W_1(c)R_2(b)R_2(c)W_3(b)$

$LDBS_2$에서는 지역 교착상태로 인하여 T_3가 철회되어 $G_1 \rightarrow G_2$의 전역 직렬성을 보장한다. [예 1] 끝

3.3 분석

GCC-M 알고리즘은 전역 직렬성을 보장하기 위하여 다음과 같은 조건을 가정하고 있다. 첫째, GTM은 전역 트랜잭션이 액세스하는 데이터 집합을 알고 있다. 둘째, LDBS는 2단계 로킹을 지원한다. 본 절에서는 각각의 가정을 만족하지 않는 경우 발생하는 문제점 및 그에 대한 해결 방안을 설명한다.

GTM에서 직접 충돌을 검사하여 이전 트랜잭션 간 직접 충돌이 없는 곳에만 직접 충돌 연산을 추가한다. 이는 전역 트랜잭션이 액세스하는 데이터 집합을 알고 있다는 가정 하에서 적용할 수 있는 것으로 만약 액세스하는 데이터 집합을 알지 못하는 경우 판독 데이터와 갱신 데이터 집합 $<RS_k^i, WS_k^i>$을 정의 할 수 없기 때문에 GTM에서 직접 충돌 유무의 검사가 어렵다. 따라서 모든 전역 트랜잭션에는 단계 2의 충돌 검사 단계는 필요하지 않고 OTM의 티켓과 같은 공통된 데이터를 사용하여 강제적인 충돌을 발생시켜야만 한다. 이 경우 OTM의 문제점을 그대로 갖지만 단계 4와 같이 서브 트랜잭션은 로크를 순서대로 획득한다면 OTM에 비해 전역 교착상태 발생 가능성은 줄일 수 있을 것이다.

[예 1]에서 동시 실행되는 전역 트랜잭션에서 OTM을 적용하는 경우 가능한 지역 스케줄은 다음 표 1과 같다. 단, 스케줄을 단순화하기 위하여 티켓 갱신 연산만 나타낸다.

경우 1과 2는 전역 직렬성을 만족하는 실행 스케줄이지만, 경우 3은 전역 교착상태가 발생하여 G_1 혹은 G_2를 철회하여 전역 직렬성을 보장한다. 반면에 GCC-M은 메시지 순서에 따라 전역 트랜잭션의 실행 순서를 사전에 제어하므로 경우 1의 실행 스케줄만 만족한다. 다음 [예 2]는 [예 1]에서 트랜잭

표 3. OTM에서 가능한 지역 스케줄

전역 스케줄	지역 스케줄
경우 1: $G_1 \rightarrow G_2$	LSH($LDBS_1$): $W_1(t_1+1)W_2(t_1+1)$ LSH($LDBS_2$): $W_1(t_2+1)W_2(t_2+1)$
경우 2: $G_2 \rightarrow G_1$	LSH($LDBS_1$): $W_2(t_1+1)W_1(t_1+1)$ LSH($LDBS_2$): $W_2(t_2+1)W_1(t_2+1)$
경우 3: $G_1 \leftrightarrow G_2$	LSH($LDBS_1$): $W_2(t_1+1)W_1(t_1+1)$ LSH($LDBS_2$): $W_1(t_2+1)W_2(t_2+1)$

션의 내용을 GTM이 모를 경우의 GCC-M 실행 과정을 보여준다.

[예 2] 전역 트랜잭션은 G_1, G_2 순으로 발생하고 OTM과 마찬가지로 직접 충돌을 유발하기 위하여 모든 서브 트랜잭션은 티켓을 액세스한다고 가정하자. $LDBS_1$의 지역 스케줄은 $G_1^1 \rightarrow G_2^1$이고 충돌 연산은 $<W_1^1(t_1+1), W_2^1(t_1+1)>$이다. 마찬가지로 $LDBS_2$의 지역 스케줄은 $G_1^2 \rightarrow G_2^2$이고 충돌 연산은 $<W_1^2(t_2+1), W_2^2(t_2+1)>$이다. GTM은 전역 트랜잭션 G_1을 액세스하는 LDBS에 따라 서브 트랜잭션으로 분할하고 G_1^1, G_1^2에 티켓 연산을 추가한 후 GTA에 전달한다. 다음 GTM은 전역 트랜잭션 G_2를 액세스하는 LDBS에 따라 서브 트랜잭션으로 분할한다. $LDBS_1$의 MOC_1에게 $[G_2^1, <W_1^1(t_1+1), W_2^1(t_1+1)>]$, $LDBS_2$의 MOC_2에게 $[G_2^2, <W_1^2(t_2+1), W_2^2(t_2+1)>]$을 각각 전달한다. 해당 정보를 수신한 MOC_1과 MOC_2는 해당 서브 트랜잭션을 실행하기 위하여 충돌 정보와 함께 LTA에게 전달한다. 그러면, $LDBS_1$의 LTA_2^1은 $W_2^1(t_1+1)$를 만날 때까지 G_2^1의 연산을 순서대로 실행 요청한다. 요청하고자 하는 연산이 $W_2^1(t_1+1)$인 경우, LTA_2^1은 먼저 MOC_1을 통해 $W_2^1(t_1+1)$와 충돌되는 연산이 실행되었는지 확인한다. 즉, 서브 트랜잭션 G_1^1의 $W_1^1(t_1+1)$의 실행 상태($S(W_1^1(t_1+1))$)를 MOC_1에게 확인하여 '1'이면 실행한다. $LDBS_2$의 LTA_2^2의 경우도 마찬가지이다. 따라서 모든 LDBS에서 $G_1 \rightarrow G_2$ 순으로 실행되기 때문에 GCC-M의 경우 OTM에 비해 전역 교착상태 발생을 줄일 수 있다. [예 2] 끝

다음으로 GCC-M은 LDBS가 2단계 로킹을 지원한다고 가정하고 있다. 그 이유는 현재 상용화된 대부분의 LDBS가 지역 동시성 제어 기법으로 2단계 로킹을 채택하고 있기 때문이다. 하지만 타임스탬프 순서기법과 낙관적 동시성 제어 기법을 지원하는 경우에도 적용이 가능하다. LDBS가 타임스탬프 순서기법을 지원하는 경우에는 2단계 로킹을 사용하는 경우에 비해 GCC-M 적용이 효과적이다. 타임스탬프 순서기법은 트랜잭션과 데이터에 타임스탬프를 할당하고 트랜잭션의 타임스탬프 순서대로 데이터를 액세스하게 하여 순서를 만족하지 못하는 경우 해당 트랜잭션을 철회함으로써 직렬성을 보장하는 기법이다. 일반적으로 타임스탬프는 트랜잭션이 LDBS에 들어오는 순서를 의미하는 것으로 메시지 순서와 일치한다. 전역 트랜잭션 T_{k-1}, T_k가 있고 메시지 순서가 $T_{k-1} \rightarrow T_k$라고 가정하자. 타임스탬프 순서기법은 전달 받은 순서대로 타임스탬프를 할당한다고 보면 $ts(T_{k-1})$는 T_{k-1}의 서브 트랜잭션에, $ts(T_k)$는 T_k의 서브 트랜잭션에 할당되고 모든 노드에서 $ts(T_{k-1}) < ts(T_k)$가 된다. 또한 타임스탬프 순서기법은 이 순서대로 트랜잭션의 실행을 보장해주기 때문에 순서대로 로킹하는 GCC-M의 단계 4는 필요없다. 따라서 모든 노드는 전달받은 서브 트랜잭션의 병행 실행이 가능하여 2단계 로킹을 사용하는 경우에 비해 동시성을 증가시킬 수 있다.

LDBS가 트랜잭션을 먼저 실행한 후 직렬성을 검증하는 낙관적 동시성 제어 기법의 경우는, 모든 노드에서 전달받은 트랜잭션 순서대로 실행하였다는 것을 보장 하다면 GCC-M을 적용하는 것이 가능하다. 즉, 낙관적 동시성 제어 기법을 사용하는 노드들은 서브 트랜잭션을 병행적으로 실행한 후 직렬성을 검증한다. 전역 트랜잭션 T_{k-1}, T_k가 있다면 낙관적 동시성 제어 기법의 경우 실행 노드 $LDBS_i$에서 모든 충돌 연산이 $T_{k-1} \rightarrow T_k$ 순으로 실행될 수 있고, $LDBS_j$에서는 $T_k \rightarrow T_{k-1}$순으로 실행될 수 있다. 이를 해결하기 위하여 2단계 로킹 적용시와 마찬가지로 원자적으로 서브 트랜잭션을 실행하는 방법과 2PC를 지원하는 GTA에서 완료 순서를 검증하는 방법이 있을 수 있다. 만약 대부분이 판독 트랜잭션이라면 전자는 낙관적 동시성 제어 기법의 장점을 활용하지 못하게 되므로 후자의 경우가 더 적합하다.

Ⅳ. 성능 평가 모형

본 절에서는 GCC-M의 성능을 기존 기법과 비교하기 위한 성능 평가 모형을 설명한다. 모의실험을 위하여 본 논문에서 개발한 분산 트랜잭션 처리 시스템 모형을 그림 2에 나타난다. 모의실험은 미국의 MCC에서 개발한 CSIM 언어[18]를 이용하여 수행

하였다.

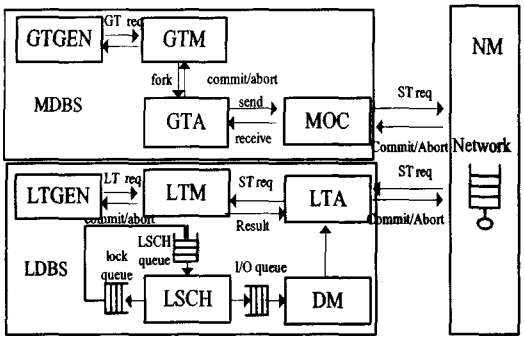


그림 2. 성능평가 모형

그림 2에서 MDBS와 LDBS는 네트워크를 통해 연결되어 있고 각 LDBS는 분산된 데이터베이스를 저장하고 있다. 먼저 MDBS 부분에 대해 설명하도록 한다. 전역 트랜잭션 생성기(Global Transaction Generator: GTGEN)는 전역 트랜잭션을 생성하는 역할을 담당한다. 전역 트랜잭션은 동시에 실행되는 여러 개의 서브 트랜잭션의 구성되는데, 각 서브 트랜잭션은 하나의 LDBS에 저장된 데이터베이스만 액세스한다. GTGEN은 여러 개의 전역 터미널로 구성된다. 각 전역 터미널은 GTM에게 전역 트랜잭션의 실행을 요청하고, 전역 트랜잭션의 실행이 완료될 때까지 대기한다.

각각의 전역 트랜잭션에 대해, GTM은 먼저 트랜잭션이 액세스하는 LDBS을 파악한 후, GTM은 TTBL와 MQ를 이용하여 충돌 검사를 실시한다. 직접 충돌 유발 작업이 끝난 후 전역 트랜잭션을 실행하기 위하여 GTM은 GTA에게 트랜잭션 내용을 전달한다. GTA는 2PC의 조정자 역할을 담당한다. GTA는 전역 트랜잭션을 구성하는 서브 트랜잭션의 실행 순서를 보장하기 위하여 MOC를 통하여 해당 LDBS로 각 서브 트랜잭션을 전송한다. MOC는 LDBS를 통합하기 위한 어댑터로 MDBS와 LDBS의 인터페이스 역할을 한다.

LDBS는 GTA로부터 요청된 서브 트랜잭션과 지역 트랜잭션 생성기(Local Transaction Generator: LTGEN)에서 생성된 지역 트랜잭션을 구분 없이 실행한다. LTA는 서브트랜잭션을 실행하는 것으로 2PC의 참여자 역할을 한다. LTGEN은 여러 개의 지역 터미널로 구성된다. 지역 트랜잭션 관리자(Local Transaction Manager: LTM)는 요청된 서브 트랜잭션과 지역 터미널에서 생성된 지역 트랜잭션을 처리하는 역할을 담당하며 이를 위해 LTM은 지역 스케줄러(Local Scheduler: LSCH) 및 데이터 관리자(Data Manager: DM)와 협조한다. LSCH는 레코드 단위의 2단계 로킹 기법을 지원한다. 각 트랜잭션에 대해 LTM은 로크 요청을 LSCH의 lsch_queue에 입력하고 LSCH는 lsch_queue에 입력된 로크 요청을 FIFO 방식으로 처리한다. 로크 요청이 허용될 경우, LSCH는 데이터 액세스 연산을 DM에게 전송한다.

본 논문에서 사용한 입력 매개 변수를 표 2에 요약한다. 각 매개 변수의 구체적인 값은 [6]과 [13]에서 많이 참조하였다.

GTM이 전체 시스템의 병목이 되는 것을 방지하기 위해서 GTM이 사용하는 CPU는 LDBS에서 사용하는 CPU보다 성능이 우수하다고 가정하였다. 전체 LDBS 노드 수는 3으로 제한하고 각 노드의 데이터베이스는 1000개의 데이터를 갖는다. 노드 당 하나의 CPU와 두 개의 디스크가 할당되며 디스크 액세스 시간은 0.01초에서 0.03초까지의 일양 분포를 따른다. CPU와 디스크는 FIFO 큐를 이용하여 I/O 요청 및 로크 요청 등을 순서대로 처리한다. 네트워크 관리자는 10Mbps의 대역폭을 갖는 FIFO 서버로 구현되었다. 네트워크를 통해 메시지를 전송하는 과정을 표현하기 위해 메시지마다 FixedMsgInst 만큼의 명령수와 ConMsgSize 만큼의 추가적인 명령수를 실행한다.

전역 트랜잭션과 지역 트랜잭션의 생성 확률은 동일하며 하나의 노드에서 실행되는 전역 트랜잭션의 부분을 서브 트랜잭션이라 하며 서브 트랜잭션의 크기는 지역 트랜잭션의 크기와 동일하다. 서브 트랜잭션과 지역 트랜잭션이 액세스하는 데이터 수는 트랜잭션 크기 ± 트랜잭션 크기 × TRSizeDev 사이의 일양분포를 따른다.

모의 실험에 사용된 주요 성능 평가 지수는 트랜잭션 처리율이다. 트랜잭션 처리율은 초당 완료되는 트랜잭션의 수로 계산하고, 동시에 실행되는 터미널 수를 증가시키면서 측정하였다. 신뢰성 있는 모의 실험 결과를 얻기 위하여 배치 평균 기법(batch mean method)을 사용한다. 즉, 본 논문에서 나타난 실험 결과들은 30개의 다른 seed를 이용하여 산출된 결과들의 평균값이다. 각각의 실험들은 완료된 트랜잭션 수가 2,000개가 될 때까지 수행하며, 초기 200개가 완료될 때까지의 결과들은 무시한다. 이러한 기법을 이용하여 산출된 결과들은 90%의 신뢰 수준을 만족하였다.

그림 2. 입력 매개 변수

시스템 구성 변수		
LCPU	노드 CPU의 속도	10MIPS
GTMCPU	GTM CPU의 속도	30MIPS
NetBandWidth	네트워크의 전송속도	100Mbps
MPL	전체 시스템에서 다중 프로그래밍 정도	5 ~ 50
NumDisk	각 노드의 디스크 수	2 disks
MinDiskTime	최소 디스크 액세스 시간	10ms
MaxDiskTime	최대 디스크 액세스 시간	30 ms
DBSize	LDBS에 저장 데이터 수	1,000
CacheHitRatio	캐쉬 히트율	80 %
오버헤드 변수 및 트랜잭션 변수		
FixedMsgInst	메시지 처리를 위한 고정 명령수	20,000
ConMsgSize	제어 메시지의 길이	256Bytes
LockInst	로크 등록/해제 명령수	300
PerIOInst	디스크 I/O 명령수	5000
LTLength	지역 트랜잭션 당 평균 액세스 데이터 수	10
STLength	전역 트랜잭션 당 평균 액세스 데이터 수	10
TRSizeDev	트랜잭션 길이의 편차	0.1
WriteOpPct	갱신 연산 비율	20%

V. 실험 결과 분석

본 절에서는 개발한 성능 평가 모형을 이용하여 수행한 실험 결과를 분석한다. 본 논문에서 구현한 전역 동시성 제어 기법은 OTM과 GCC-M이다. 다양한 데이터베이스 액세스 유형[8]에 따른 전역 동시성 제어 기법의 성능을 분석하기 위하여 데이터베이스 액세스 유형을 높은 충돌 환경, 분할 액세스 환경, 그리고 균일 액세스 환경으로 나누어 실험을 수행하였다.

5.1 높은 충돌 환경

높은 충돌 환경은 모든 노드에서 실행되는 트랜잭션들이 데이터베이스의 특정 부분을 높은 확률로 액세스하는 경우를 모델링한 것이다. 이 환경에서 각 트랜잭션이 실행하는 연산의 80%는 전체 데이터베이스의 특정 20%(200개)를 액세스한다. 트랜잭션의 나머지 20%의 연산은 나머지 데이터베이스를 무작위로 액세스한다.

그림 3은 터미널 수를 다양하게 변화시킬 때, 각 전역 동시성 제어 기법의 성능 변화를 나타낸다. 전체 시스템의 터미널 수(MPL)는 5에서 50까지 변화한다. 트랜잭션의 평균 갱신 비율은 20%로 두었다. 터미널 수가 증가할수록 GCC-M의 성능이 OTM에 비해 우수해지며, 터미널 수가 50인 경우는 OTM에 비해 2배 정도의 성능 향상을 보인다. 그 이유는 다음과 같다. OTM은 전역 트랜잭션은 어떤 제약 사항 없이 동시에 실행되고 동일한 데이터인 티켓을 액세스한다. 따라서 OTM은 터미널 수가 증가할수록 동시에 실행되는 전역 트랜잭션 수가 증가하므로 빈번한 데이터 충돌과 전역 교착상태로 인하여 많은 트랜잭션이 철회된다. 반면에 GCC-M의 경우 전역 트랜잭션의 T_k의 이전 트랜잭션 T_{k-1}의 충돌 연산이 실행되지 않은 경우 실행을 지연시키므로, 전역 트랜잭션의 동시성이 제한된다. 게다가 모든 LDBS는 전송된 순서대로 전역 트랜잭션을 실행하므로 전역 교착상태가 발생하지 않는다. 따라서 동시성이 증가할수록 OTM은 급격한 성능 저하 현상을 보인다. 그로 인하여 터미널 수 30부터는 OTM의 전역 트랜잭션 처리율은 거의 0에 가까워진다. 그 이유는 대부분의 전역 트랜잭션이 교착상태로 철회되기 때문이다.

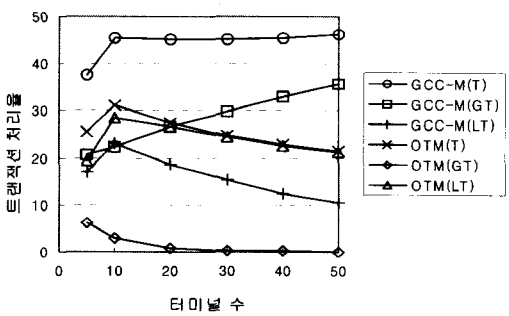


그림 3. 높은 충돌 환경에서 터미널 수 변화에 따른 트랜잭션 처리율

전반적으로 전역 트랜잭션인 경우 OTM의 성능이 GCC-M보다 떨어지지만, 지역 트랜잭션인 경우 OTM의 성능이 우수하다. 그 이유는 터미널 수가 증가할수록 대부분의 전역 트랜잭션이 티켓으로 인한 로킹 대기와 철회로 인해 처리가 지연되는 반면에 지역 트랜잭션의 경우 상대적으로 데이터 충돌율이 줄어들기 때문이다.

5.2 분할 액세스 환경

분할 액세스 환경은 각 노드마다 액세스하는 데

이터들이 분할되어 있고, 노드는 자신에게 할당된 데이터를 주로 액세스하는 환경을 모델링한 것이다. 각 트랜잭션이 실행하는 연산의 20%는 시스템 카탈로그나 파일 디렉토리, 혹은 B-트리 인덱스의 루트 노드나 중간 노드와 같은 공유 데이터를 액세하며, 80%의 연산은 해당 노드에 특정 분할된 데이터를 액세스한다. 본 데이터 액세스 모델은 다른 전역 트랜잭션에 의해 동일한 데이터를 액세스할 확률이 낮기 때문에 데이터 충돌율이 상당히 줄어든다.

그림 4는 터미널 수를 다양하게 변화시킬 때, 각 전역 동시성 제어 기법의 성능 변화를 나타낸다. 트랜잭션의 평균 갱신 비율은 20%로 두었다. 예측대로 낮은 데이터 충돌 확률과 로킹 지연으로 인하여 높은 충돌 환경보다 모든 기법의 성능이 향상되었다. 특히 OTM의 성능이 GCC-M의 보다 많이 향상되었다. 그 결과 터미널 수가 증가할수록 GCC-M과의 성능차이 20% 정도로 줄어들었다. 그 이유는 다음과 같다. 첫째, 데이터 충돌을 줄여 교착상태 발생가능성을 줄어들게 했다. 특히 전역 트랜잭션의 서브 트랜잭션과 지역 트랜잭션의 충돌 확률이 줄어들게 된다. 이것은 OTM의 지역 트랜잭션의 성능이 상당히 향상되었으므로 확인할 수 있다. 둘째, 높은 충돌 환경에 비하여, GCC-M은 서브 트랜잭션들 사이의 직접 충돌 확률이 줄어들기 때문에 강제적인 충돌 연산 추가 가능성이 높다. 따라서 서브트랜잭션과 지역 트랜잭션 사이의 로킹 충돌 가능성이 높아지게 된다.

높은 충돌 환경에 비하여 GCC-M과 OTM의 전역 트랜잭션 처리율은 성능 차이가 크지 않다. 그 이유는 다음과 같다. 두 기법 모두 전역 직렬성을 보장하기 위하여 강제적인 충돌 유발 연산을 이용한다. 분할 액세스 환경에서는 각 서브 트랜잭션간의 충돌 발생률이 적다. 그러므로 GCC-M의 경우에도 대부분의 서브 트랜잭션이 추가적인 연산 오버헤드를 갖게된다. 결과적으로 직접 충돌이 없는 경우에만 강제적인 충돌을 유발하는 GCC-M의 장점이 상쇄된다. OTM은 티켓으로 인한 교착상태 발생율이 여전히 높기 때문에 터미널 수가 증가하면 전역 트랜잭션 처리율은 거의 0에 가깝다.

5.3 균일 액세스 환경

균일 액세스 환경은 데이터베이스에 저장된 모든 데이터를 동일한 확률로 액세스하는 환경을 모델링한 것이다. 이 환경에서는 각 트랜잭션이 실행하는 연산의 10%는 공유 데이터를 액세스하며, 나머지

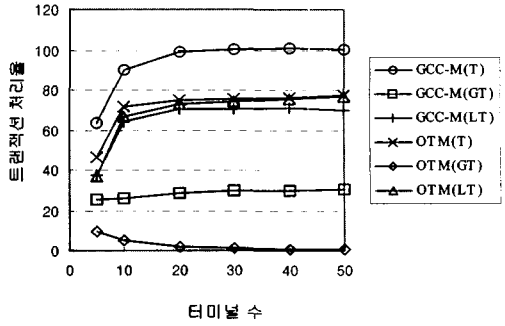


그림 4. 분할 액세스 환경에서 터미널 수 변화에 따른 트랜잭션 처리율

90%는 데이터베이스의 나머지 부분을 균일하게 액세스한다. 분할 액세스 환경과 유사하게 공유 데이터 부분은 전체 데이터베이스의 20%를 차지한다. 그림 5는 갱신 연산 비율을 20%로 두고 터미널 수를 5에서 50까지 변화하면서 실험한 결과이다.

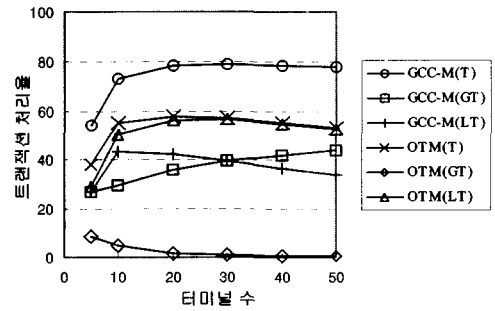


그림 5. 균일 환경에서 터미널 수 변화에 따른 트랜잭션 처리율

분할 액세스 환경과 마찬가지로 높은 충돌 환경에 비해 모든 기법의 트랜잭션 처리율이 높아진다. 그러나 분할 액세스 환경에 비해 성능이 낮다. 성능 향상 정도가 높은 충돌 환경보다 낮고 분할 액세스 환경보다 높은 것은 균일 액세스 환경에서 동일한 데이터를 액세스할 확률이 높은 충돌 환경에 비해서 낮고 분할 액세스 환경에 비해 높기 때문이다. 그 결과 터미널 수가 증가할수록 GCC-M와 OTM의 성능 차이는 30% 정도이다. 이것은 분할 액세스 환경보다는 높고 높은 충돌 환경보다는 낮다.

VI. 결 론

본 논문에서는 MDBS를 위한 메시지 순서 기능을 적용한 전역 동시성 제어 기법인 GCC-M을 제

안하였다. 메시지 순서 기능은 하나의 노드가 여러 개의 메시지를 송신하는 경우, 모든 노드는 송신한 순서에 따라 메시지 수신을 보장하는 것으로 이를 이용하여 전역 트랜잭션의 서브 트랜잭션들을 실행 노드에 같은 순서로 전송하고 실행 LDBS는 해당 순서대로 직렬화한다면 전역 직렬성을 보장할 수 있고 전역 교착상태를 방지할 수 있다. 그 결과 GCC-M은 LDBS의 실행 정보 없이 전역 트랜잭션의 상대적인 실행 순서를 결정할 수 있기 때문에 지역 자치성을 보장한다.

GCC-M을 기존의 지역 자치성 보장 기법인 OTM과 비교하기 위하여 모의실험 모형을 개발하였고, 다양한 데이터 액세스 환경에서 실험을 하였다. 실험 결과를 정리하면 다음과 같다. 첫째, GCC-M은 동시에 실행되는 터미널 수가 증가함에 따라 데이터 액세스 환경에 관계없이 OTM에 비해 성능이 향상됨을 보인다. 그 이유는 메시지 순서를 전역 트랜잭션의 실행 순서로 간주하여 서브 트랜잭션간에 직접 충돌이 있는 경우에는 강제적인 충돌 연산을 추가 하지 않았기 때문에 모든 전역 트랜잭션들이 공통된 데이터를 액세스하는 OTM에 비해 데이터 충돌 확률을 줄였기 때문이다. 둘째, 높은 충돌 환경에 비하여 분할 액세스 환경에서 GCC-M의 전역 트랜잭션 처리율 성능 향상 정도가 높지 않다. 분할 액세스 환경에서는 각 서브 트랜잭션간의 충돌 발생률이 적고 GCC-M은 대부분의 서브 트랜잭션이 충돌 유발을 위한 추가 오버헤드를 가지므로 결과적으로 직접 충돌이 없는 경우에만 강제적인 충돌을 유발하는 GCC-M의 장점이 상쇄되기 때문이다.

본 논문의 향후 연구 과제로는 전역 트랜잭션이 액세스하는 데이터 집합을 GTM이 알지 못하는 환경으로 GCC-M을 확장하는 방법과 실제 시스템에서 GCC-M의 구현 및 성능 평가 등을 들 수 있다. 특히, GCC-M의 구현을 위하여 그룹통신 메시징 시스템과 분산 데이터베이스 시스템이 모두 필요하므로, 본 연구의 관련 연구로 수행되고 있는 그룹통신 메시징 시스템의 시제품에 분산 데이터베이스 시스템의 기능 일부를 통합하는 작업이 현재 계획 중에 있다.

참 고 문 헌

[1] J. Albert, "Theoretical Foundations of Scheme Restructuring in Heterogeneous Multidata- base Systems," *Proc. CKIM*, pp. 461-470, 2000

[2] A. Brayner and T. Härder, "Global Semantic Serializability: An Approach to Increase Concurrency in Multidatabase Systems," *Proc. CoopIS*, pp. 301-315, 2001

[3] Y. Breitbart, H. Garcia-Molina and A. Silberschatz, "Overview of Multidatabase Transaction Management," *VLDB Journal*, 1(2), pp. 72-79, 1992

[4] Y. Breitbart, D. Georgakopoulos, M. Rusinkiewicz and A. Silberschatz, "On Rigorous Transaction Scheduling," *IEEE Trans. on Software Eng.*, 17(9), pp. 954-960, 1991

[5] Y. Breitbart and A. Silberschatz, "Multidatabase Update Issues," *Proc. ACM SIGMOD*, pp. 135-142, 1988

[6] M. Carey and M. Livny, "Distributed Concurrency Control Performance: A Study of Algorithms, Distribution, and Replication," *Proc. VLDB*, pp. 13-25, 1988

[7] D. Chappell and R. Monson-Haefel, "Guaranteed Messaging with JMS," *Java Developer's Journal*, April, 2001

[8] H. Cho, "Performance of Cache Coherency Schemes in a Shared Disks Transaction Environment," *Proc. IEEE Workshop on Future Trends of Distributed Computing Syst.*, pp. 154 -161, 1997

[9] W. Du, A. Elmagarmid and W. Kim, "Maintaining Quasi Serializability in Multidatabase," *Proc. Research Issues in Data Eng.*, pp. 360-367, 1991

[10] Fiorano, "FioranoMQ and Progress SonicMQ Highlights," *http://www.fiorano.com*, 2001.

[11] D. Georgakopoulos, M. Rusinkiewicz and A. Sheth, "Using Tickets to Enforce the Serializability of Multidatabase Transactions," *IEEE Trans. on Knowledge and Data Eng.*, 6(1), pp. 166-180, 1994

[12] J. Holliday, D. Agrawal and A. Abbadi, "Using Multicast Communication to Reduce Deadlock in Replicated Databases," *Proc. IEEE Symposium on Reliable Distributed Syst.*, pp. 196-205, 2000

[13] B. Kemme and G. Alonso, "A New Approach to Developing and Implementing

Eager Database Replication Protocols," *ACM Trans. Database Syst.*, 25(3), pp. 333-379, 2000

[14] P. Laumay, E. Brueton, N. de Palma and S. Krakowiak, "Preserving Causality in Scalable Message-oriented Middleware," *Lecture Notes in Computer Science*(2218), pp. 311-328, 2001

[15] S. Lee, C. Hwang and W. Lee, "A Uniform Approach to Global Concurrency Control and Recovery in Multidatabase Environment," *Proc. CIKM*, pp. 51-58, 1997

[16] K. Lee and S. Park, "Stable Transaction Management for Preserving the Global Integrity Constraints in Multidatabase Systems," *Proc. CoopIS*, pp. 396-405, 1998

[17] S. Mehrotra, R. Rastogi, H. F. Korth and A. Silberschatz, "Ensuring Consistency in Multidatabases by Preserving Two-Level Serializability," *ACM Trans. Database Syst.*, 23(2), pp. 199-230, 1998

[18] H. Schwetman, *CSIM Users Guide for use with C Revision 16*, MCC, 1992

[19] Sonic, "Getting Started with SonicMQ V4," *http://www.sonicsoftware.com*, 2001

[20] A. Tannebaum and M. van Steen, *Distributed Systems Principles and Paradigms*, Prientice Hall, 2002

[21] Y.C. Tay, N. Goodman and R. Suri, "Locking Performance in Centralized Databases," *ACM Trans. Database Syst.*, 10(4), pp. 415-462, 1985

[22] M. Wisemann, X. Défago and A. Schiper, "Group Communication based on Standard Interfaces," *Proc. IEEE Symp. Network Computing and Applications*, pp. 140-147, 2003

[23] A. Zhang and A. Elmagarmid, "A Theory of Global Concurrency Control in Multidatabase Systems," *VLDB Journal*, 2(3), pp. 331-360, 1993

문 애 경 (Aekyung Moon) 정회원

1992년 2월 : 영남대학교 전산 공학과 학사
1997년 2월 : 영남대학교 컴퓨터 공학과 석사
2000년 2월 : 영남대학교 컴퓨터 공학과 박사
1995년~현재 : 영남대학교
2000년 4월~현재 : 한국전자통신연구원 소프트웨어 로봇 연구팀 선임연구원

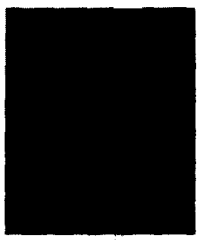
<관심분야> 분산/병렬 데이타베이스, 메시징시스템

남 궁 한(Han Namgoong) 정회원

1982년 1월 : Columbia 대학 전산학과 석사
1982년 ~ 1986년 : LG전자 컴퓨터 사업부
1987년 2월 ~ 현재 : 한국전자통신연구원 책임연구원

<관심분야> 분산시스템, fault-tolerant system, group communication

조 행 래(Haengrae Cho) 정회원



1988년 2월 : 서울대학교 컴퓨터공학과 학사
1990년 2월 : 한국과학기술원 전산학과 석사
1995년 2월 : 한국과학기술원 전산학과 박사

1995년~현재 : 영남대학교 전자정보공학부 부교수

<관심분야> 분산/병렬 데이터베이스, 트랜잭션 처리, DBMS 개발